Week 8 Designing Classes

In this exercise you will design the classes necessary to implement a simplified music playing program that can play digital music files such as mp3s. The purpose of this exercise is primarily to help you think about decomposing a problem into an appropriate set of classes that you can implement.

Consider the problem description below. Design the set of classes necessary to implement code solving this problem. As part of your design, include the methods that each class needs and the internal data members that must be used. For this assignment, you do not need to implement the methods, but you should have a clear idea of *how* you would implement the methods you identify.

A music player plays music from a playlist of songs. A playlist is a list containing anywhere between 1 and 200 (inclusive) songs. Each song has the following pieces of information: the song's title, the artist, the album name, and the filename of the song on the computer's disk. In addition, each song is encoded into one of the following formats determined by the filename extension: 'mp3', 'ogg', 'wav', or 'wma'.

Each song in a playlist can be identified by either its title or by its index in the list (e.g., the first song is at index 0, the second song at index 1, and so forth). Songs may be added to a playlist by providing the filename of the song stored on the computer's disk. A song can be removed from the playlist by specifying either its index or its title. A playlist must be able to randomize the order of the songs in the list, and a playlist must provide a method to print all of its songs to the screen.

With regard to accessing songs, a playlist must provide mechanisms to return the first song in its list, return the next song (after the currently playing one) in the list, and return the i_th song as specified by the user. Note that once the last song in a playlist is returned, the next song to be returned is the first song in the list.

A music player's purpose is to play songs from a playlist. When given a playlist, a user can play music by selecting any song in the playlist (according to its index), and the music player will play that song and then continue to play all songs after it. If the end of the playlist is reached, the music player should wrap around and begin playing at the beginning of the list. If the user does not provide a specific starting point, the music player should start playback at the beginning of the list.

For simplicity, you may assume that there is a utility class called AudioPlayer with methods playMP3(), playOgg(), playWave(), and playWMA() that each accept a song as a parameter and plays the song. These functions return when the song has finished (thus, there is no concept of "fast forward", "rewind", or "stop" in this player; once the player has started it keeps on going).

Hints and Questions:
1. Think about the nouns in the description. What are the things that the description requires you to create?
2. What kinds of operations should you provide for each of these things?
3. How do these things interact with each other?
4. Who is/are the user(s) of each class you define?
5. Which, if any, of your methods and fields are static? Why?
6. Are any of your classes immutable? Which ones?

If you've completed a class design to your satisfaction, write pseudo-code describing the behavior of the "main loop" of your music player. That is, write pseudocode describing how the player walks through the playlist and plays music.