Capstone Project for C# Programming Certificate

The application you will build is CD Catalog that will allow users to store and retrieve information about their collection of songs/CDs.

### Main Interface

As the program starts, the user will be shown several options:

1. Add a new Song
2. Add a new CD
3. Rate a specific song or CD
4. Find a CD or song by title
5. Find CDs or songs by a specific artist
6. Find CDs or songs within a specific genre
7. Create a “playlist” for specific time segment

The minimum classes required for the program will be as follows:

* Song
  + Title
  + Artist
  + Album (link to Album)
  + Track Number
  + Genre
  + Track Length
  + Rating
* Album
  + Title
  + Year
  + Rating
  + Artist
* Artist
  + Name

### Menu Items

The first two options allow you to enter information about a song or CD. Entering information about a song allows you to include songs that were not purchased as part of a CD (the CD name is still required however). Entering songs by CD is more efficient than song-by-song, as it allows you to specify the CD information once and not with every song. Both options allow you to specify the information about the song and CD, and make the appropriate Song and Album records (a CD with 12 songs will result in 12 Song records and 1 CD record).

The Rate a Specific Song or CD option, of course, allows you to rate a song or CD. The option asks the user for the name of the song/album, then the rating. If more than one song/CD fits the name specified, list all matching items (show if it’s a CD or Song, the artist, and – if it’s a song – the album it’s on) and provide a mechanism that allows the user to specify which matching item they want to rate. Only one item can be rated at a time.

Rating song or album allows the user to associate a 1-5 star rating based on how much they like the song/album. When displaying songs individually (not by album), they should be sorted by ratings (highest displayed first). When displaying albums, the list should be sorted by the album rating (highest displayed first).

The next 3 menu options allow you to search for songs or albums based on title, artist, or genre. Searches should be general, meaning if the title specified by the user fits all or part of the song/CD title, it should be displayed. This listing should show 1) if it’s a song or CD, 2) the full title, and 3) the artist. From this list, the user can select any item listed and display more detailed information about the item. In the case of song items, display the Title, Artist, Album title, Track Number, Genre, Track Length, and Rating. In the case of CDs, display the Title, Artist, Year, and Rating, followed by a list of all the songs on the CD (that are in the collection) by track number.

The final option allows the user to specify the amount of time they want to have music play (e.g. 45 minutes), and you will create a random playlist that will contain a list of songs that, together, will last 45 minutes (within a 1 minute margin, so 44-46 minutes). List the songs with the title, artist, album, and time of the song. This time is to be filled with individual songs (i.e. don’t consider albums in this process, just individual songs).

### Data Storage

Information about the books may be kept in memory while the program is running. As the program exists, this information should be saved in a method of your choice. Some obvious options are serialization, database, XML document, or text document(s). The program should load this information automatically as the program starts.

Notes:

1. The program can be written as either a console application or a Windows Forms application (WinForms or WPF).
2. The system should not allow you to add an song/CD that is already in the system, but should allow duplicate songs if they are on separate CDs.
3. The various collections of data (songs, albums, artists) may be stored internally using any mechanism you choose (arrays, lists, etc). Although remember that this data must be saved to the disk somehow as the program exits and read in as the program starts. The program must be designed to hold a minimum of 1000 books, 1000 authors, 1000 subjects, and 20 employees.
4. Editing existing songs/albums is not a requirement, but may be added if the student wishes the challenge and a more full-featured example of their work. The student will not be penalized for errors in features that are beyond the base requirements.
5. All expected errors are to be handled via exception handling.
6. Remember to consider this an example project to show potential employers. As such, feel free to use it as a vehicle to show more advanced concepts (e.g. interfaces, operator overloading, delegates, multi-threading, etc.).
7. Code files should be laid out according to standard C# conventions (i.e. each class in its own source file).
8. The code should be well commented. If a beginning programmer can’t understand what is going on in your code, you need more comments!

## Testing steps:

As I evaluate the program at the end of the project, I will go through the following steps:

* Verify that the basic display and functionality requirements in the above document were met, and that all required components are complete and functional.
* I will evaluate the program based on expected scenarios:
  + I will enter several new song records.
  + I will enter one or more new CDs.
  + I will verify that the system can retrieve and display those records successfully by searching for titles, artists, and genres.
  + I will rate several songs and CDs.
  + I will try searching for titles that produce multiple entries to verify that the general list and details screens both work and appear as described above.
  + I will exit the program and restart it, verifying that the data has not been lost.
  + I will have the song create one or more playlists, and verify that it does so according to the specifications above.
* I will evaluate the program based on unexpected scenarios:
  + I will attempt to look up items that do not exist. An appropriate error message should be displayed
  + I will attempt to add songs/CDs that already exist, and verify that the program recovers and a proper error message is displayed
* I will then exit the program and look at the code, and verify that:
  + The classes are defined according to the above instructions (properties, access modifiers, etc.)
  + Verify that the code is laid out properly in appropriate files and is well commented.

If the program passes the above tests and evaluations, the student will pass the evaluation.