**Playlist**

Song

* [**Song.java**](https://drive.google.com/file/d/1Mr6U_54DH35qWomdBzZM6bZWNDBcxLHC/view?usp=sharing)
* [**SongRunner.java**](https://drive.google.com/file/d/1psD-h1FhayCw8Wj8p_f1irW3Oobc-A3A/view?usp=sharing)

Album

* [**PlayList.java**](https://drive.google.com/file/d/1FJGDOBuEaO5RPaZGoxj6pQeFFhaKsY2I/view?usp=sharing)
* [**PlayListRunner.java**](https://drive.google.com/file/d/1djdCB6mDdVcKa1tTzAfkAjvoqwwUgZy7/view?usp=sharing)

[PlayList.javaPlayList.java](https://drive.google.com/file/d/1FJGDOBuEaO5RPaZGoxj6pQeFFhaKsY2I/view?usp=sharing)

**Song Class**

You will complete the existing **Song.java** file which contains some method stubs for you. When you are finished you will need the following completed:

* Instance Variables
  + String title
  + String album
  + String artist
  + int songLength
* Two Constructors
  + default constructor used to set the instance variables to default values
  + initialization constructor used to set the instance variables (in the order given above)
* Accessor and Modifier Methods for each Instance Variable
* equals() Method
  + Add an equals method to check for equality between two Song objects. It should compare the title, album, artist and length when determining if the song is equal to another song.
* toString() Method returns string with instance variable values in the following format title/album/artist(songLength)

When you have completed Song.java, you can use **SongRunner.java** to test your program. This is a completed file to help you test your program.

Here is the expected output after running **SongRunner.java**

**//(0)**

**title/album/artist(333)**

**newTitle**

**newAlbum**

**newArtist**

**500**

**false**

**true**

**true**

**PlayList**

You will complete the PlayList.java file which contains some method stubs for you. When you are finished you will need the following completed:

* Instance Variable (should be private)
  + Song[] list
* Two Constructors
  + default constructor used to set the instance variables to default values - a playlist containing 1 song created with the Song default constructor.
  + initialization constructor used to set the instance variable - a playlist with x songs none of which are initialized.
* Accessor and Modifier Methods for Instance Variable
* addSong(int x, Song s)
  + method to add a song to the playlist at the position indicated
    - the first song in the playlist is at x = 0
    - if there is a null reference at x, add the song at location x
    - if there is a song at x, shift everything at position x and beyond by 1 position and insert song at location x (i.e. makes the array one position larger).
    - if x is larger then the last array position, insert the song at the end of the playlist (i.e. makes the array one position larger).
* getSong()
  + method to get a song from the playlist
* public int numSongs()
  + method to get the number of songs in the playlist
* totalLength()
  + method to get the total length of all the songs in the playlist
* removeArtist()
  + method to remove all the songs from the specified artist from the playlist. Your new playlist should be shorter with only the songs by that artist removed.
* removeLength()
  + method to remove all the songs that are longer than the length specified from the playlist
* shuffle()
  + method to shuffle the order of the playlist - use the card shuffling example from the arrays presentation to help
* equals()
  + I am providing you the equals method
* toString()
  + I am providing you the toString() method

When you have completed PlayList.java, you can use **PlayListRunner.java** to test your program. This is a completed file to help you test your program.

The expected output after running **PlayListRunner.java** is (**shuffled** - order may vary)

**[//(0)]**

**[null, null]**

**[null, null]**

**song1/album1/artist1(250)**

**2**

**600**

**[song2/album2/artist2(350)]**

**[song1/album1/artist1(250)]**

**[song1/album1/artist1(250), song2/album2/artist2(350)]**

**[null, null]**

**[song1/album1/artist1(250), song2/album2/artist2(350)]**

**[song1/album1/artist1(250), song2/album2/artist2(350)]**

**false**

**true**

 **PlayList.zip** (containing Song.java & PlayList.java)