Music Playlist Iteration 01

You're building a tiny console app that loads songs from a CSV file, shows the library list of songs, lets a listener play a specific song by index, and lets them queue songs to play next using your own singly linked list (not Java's LinkedList).

Requirements: User Stories and Acceptance Criteria

User Story 1 — Load my music

As a listener, I want to load songs from a CSV file so I can work with my playlist.
Acceptance Criteria

- Ask for a file and reads title, artist, durationSeconds.

```
Example File data: Note II,Wojciech Golczewski,432
```

- If a line has an error the app prints a short message and continues reading from file
  Line (actual line number) skipped: reason
- After reading, the app prints how many songs were loaded.

Example Menu Choice: 1. Load Songs from CSV

```
Enter your choice (1–7):  1
Enter CSV filename: file.csv
Could not open file: file.csv
No songs were loaded


Enter your choice (1–7): 1
Enter CSV filename: bad-list.csv
Line 2 skipped: bad number.
Line 3 skipped: invalid values.
Line 4 skipped: invalid values.
Line 5 skipped: invalid values.
Line 7 skipped: bad number.
Loaded 2 songs.
```

User Story 2 — See what’s in the playlist

As a listener, I want to see all the songs with their numbers so I can choose one to play.

Acceptance Criteria

- The app prints one song per line in the format:
  [index] "Title" by Artist (m:ss)
  - Indices displayed in the playlist and used for user input are **0-based** (the first song is index 0).
  - Duration should display in the format m:ss, where seconds are zero-padded to two digits. Example: 245 seconds will be converted to 4:05
- If no songs are loaded, it prints “Playlist is empty.”

Example Output Menu Choice: 2. Display Playlist

```
Enter your choice (1–7): 2
[0] "3.4_1-runtime-error.rx2" by Mac Quayle (3:16)
[1] "Command 64" by Wojciech Golczewski (2:22)
[2] "3.4_1-runtime-error.rx2" by Mac Quayle (3:16)
[3] "Command 64" by Wojciech Golczewski (2:22)
```

User Story 3 — Play a specific song now

As a listener, I want to play a song by its number so I can hear exactly what I want.

Acceptance Criteria

- The app prompts for an index (the number shown in the list).
- If the index is valid, the app prints: Now playing: "Title" by Artist (m:ss).
- If the index is out of range or not a number, it prints a clear message: “Invalid index.” Repeats questions until valid value based on how many songs are in the play list

Example Output Menu Choice: 3. Play a Song by Index

```
Enter your choice (1–7): 3
Enter index to play: 8
Please enter a number between 0 and 6.
Enter index to play: 5
Now playing: "Command 64" by Wojciech Golczewski (2:22)
```

User Story 4 — Add songs to the Up-Next list

As a listener, I want to add a song to an Up-Next list so my chosen tracks will play in order.

Acceptance Criteria

- The app prompts for an index and, if valid, adds that song to the end of the Up-Next queue (your custom singly linked list).
- If the index is invalid, it prints “Invalid index. Nothing enqueued.”
- (No duplicate checks are required in this iteration.)

Example Output Menu Choice: 4. Add Song to Up-Next Queue

```
Enter your choice (1–7): 4
Enter song number to add to Up-Next List (queue): 8
Please enter a number between 0 and 6.
Enter song number to add to Up-Next List (queue): 0
Song added to Up-Next.
```

User Story 5 — See what’s coming up

As a listener, I want to see the Up-Next list so I know what will play after the current song.

Acceptance Criteria

- The app prints each queued song in order, one per line:
  [position] "Title" by Artist (m:ss)
- If the queue is empty, it prints “Queue is empty.”

Example Output Menu Choice: 5. Show Up-Next Queue

```
Enter your choice (1–7): 5
[0] "3.4_1-runtime-error.rx2" by Mac Quayle (3:16)
[1] "1.8_4-oneconstant.caf" by Mac Quayle (3:31)
[2] "Uncharted Worlds" by Sam Hulick (1:15)

Now playing: "3.4_1-runtime-error.rx2" by Mac Quayle (3:16)
```

User Story 6 — Play the next song in line

As a listener, I want to play the next song from Up-Next so the queue moves forward automatically.

Acceptance Criteria

- If Up-Next has songs, the app removes the first one from the queue and prints:
  Now playing: "Title" by Artist (m:ss)
- If the queue is empty, it prints “Queue is empty.”

Example Output Menu Choice: 6. Play Next from Queue

```
Now playing: "3.4_1-runtime-error.rx2" by Mac Quayle (3:16)
```

User Story 7 - Provide Menu to Make Selection

As a listener, I want a simple numeric menu so I can make my selection.
Acceptance Criteria

Displays Menu	Matching User Story
=== Music Playlist Menu === 1. Load Songs from CSV 2. Display Playlist 3. Play a Song by Index 4. Add Song to Up-Next Queue 5. Show Up-Next Queue 6. Play Next from Queue 7. Exit	1. User Story 1 — Load my music 2. User Story 2 — See what’s in the playlist library 3. User Story 3 — Play a specific song now 4. User Story 4— Add songs to the Up-Next queue 5. User Story 5 — See what’s coming up 6. User Story 6 — Play the next song in line 7. Exit

Unified Modeling Language (UML) Diagram

Song
-name: String -artist: String // -genre: String // -releaseYear: int -durationSeconds: int -durationMinutes: String
+ Song(name, artist, genre, releaseYear, durationSeconds) + getName(), getArtist(), getDuration(), //getGenre(),

//getReleaseYear() + toString()

CSV Import / Loader
-lineNumber: int
+loadFromCSV() (Return null if invalid) +parseSongLine() -name: String -artist: String -durationText: String

Playlist / Library
- SongList<Song>: ArrayList - SongQueue<Song>: **Singly LinkedList - playlistName: String

+ MusicPlaylist(): constructor
+ getName()
+ getSize()
+ showAllSongs()
+ addSong()
+ playSong()

Queue
-head: Node<T> -tail: Node<T> -count: int
+ SinglyLinkedList() + size(): int + isEmpty(): boolean + addLast(value: T): void + removeFirst(): T + displayAll(): void

Test Cases for Test-Driven Development (TDD):

User Story 1:

[redacted]

Case	Pre-Condition	Post-Condition / Output
Valid csv line added	Get It,Keshi,151	[0] "Get It" by Keshi (2:31)
Invalid data type in one of the fields (String, String, int) Reject input, continue with rest of csv file. Print specific error message.	name,artist,durationSeconds	Line 1: invalid duration "durationSeconds" → skipping line.
Incorrect amount of data fields on line Reject input, continue with rest of csv file. Print specific error message.	Missing, entryline	Line 32: wrong number of fields → Missing, entryline

```
Enter CSV filename: Project_01.csv
Line 1: invalid duration "durationSeconds" → skipping line.
Line 32: wrong number of fields → Missing, entryline
Loading complete. Loaded 30 songs.
[0] "Get It" by Keshi (2:31)
```

User Story 2:

Case	Pre-Condition	Post-Condition / Output

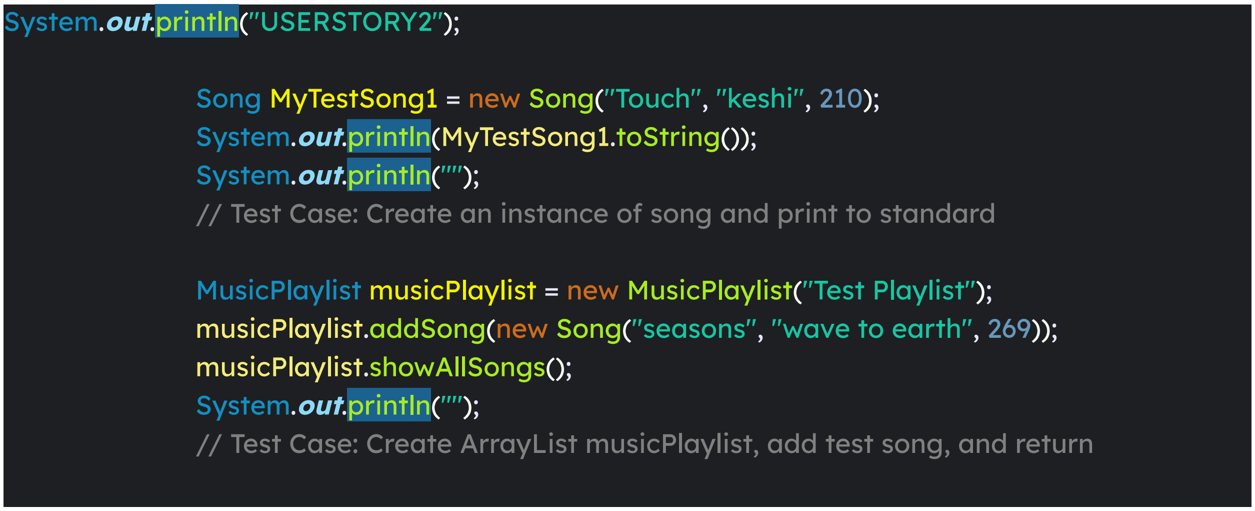
Create an instance of a valid song and print	new Song("Touch", "keshi", 210);	"Touch" by keshi (3:30)

Create a playlist, add a song directly, and print to format (including index and seconds -> minutes conversion)	musicPlaylist.addSong (new Song("seasons", "wave to earth", 269));
	musicPlaylist.showAllSongs();
		[0] "seasons" by wave to earth (4:29)

Add a song to playlist as an object and print (test multiple songs printing)	^^ ++ vv
	Song MyTestSong2 = new Song("summer", "keshi", 165);
	musicPlaylist.addSong (MyTestSong2);
	musicPlaylist.showAllSongs();
		[0] "seasons" by wave to earth (4:29)
		[1] "summer" by keshi (2:45)

Create an empty playlist and return songs (empty playlist statement)	MusicPlaylist musicPlaylist2 = new MusicPlaylist("Empty Test Playlist");
	musicPlaylist2.showAll Songs();
		“Playlist is empty.”

Song is 0 length (test duration conversion)	new Song("summer", "keshi", 0);	[1] "summer" by keshi (0:00)



```
System.out.println("USERSTORY2");

        Song MyTestSong1 = new Song("Touch", "keshi", 210);
        System.out.println(MyTestSong1.toString());
        System.out.println("");
        // Test Case: Create an instance of song and print to standard

        MusicPlaylist musicPlaylist = new MusicPlaylist("Test Playlist");
        musicPlaylist.addSong(new Song("seasons", "wave to earth", 269));
        musicPlaylist.showAllSongs();
        System.out.println("");
        // Test Case: Create ArrayList musicPlaylist, add test song, and return
        Song MyTestSong2 = new Song("summer", "keshi", 165);
        musicPlaylist.addSong(MyTestSong2);
        musicPlaylist.showAllSongs();
        System.out.println("");
        // Test Case: Add song as object

        MusicPlaylist musicPlaylist2 = new MusicPlaylist("Empty Test Playlist");
        musicPlaylist2.showAllSongs();
        System.out.println("");
        // Test Case: Empty playlist

        musicPlaylist2.addSong(new Song("seasons", "wave to earth", 0));
        musicPlaylist2.showAllSongs();
        System.out.println("");
        // Test Case: Funky second to minute duration conversion
```

User Story 3:

Case	Pre-Condition	Post-Condition / Output
Input next song index within playlist	Enter your choice (0-length of arraylist - 1) Input: 0	Now playing: "seasons" by wave to earth (4:29)
Input next song index outside of playlist bounds (less than 0, greater than the length size). Reprompt.	Please enter a number between 0 and (length of arraylist - 1) Input: 5	Invalid index Please enter a number between 0 and 1
Input improper data type for index. Reject and reprompt.	Please enter a number between 0 and 1 Input: one	Invalid index Please enter a number between 0 and 1

```
System.out.println("USERSTORY3");

        musicPlaylist.showAllSongs();
        musicPlaylist.playSong();
        // Call method to sandbox different input
```

User Story 4:

[redacted]

Case	Pre-Condition	Post-Condition / Output
Valid csv line added	Get It,Keshi,151	[0] "Get It" by Keshi (2:31)
Invalid data type in one of the fields (String, String, int) Reject input, continue with rest of csv file. Print specific error message.	name,artist,durationSeconds	Line 1: invalid duration "durationSeconds" → skipping line.
Incorrect amount of data fields on line Reject input, continue with rest of csv file. Print specific error message.	Missing, entryline	Line 32: wrong number of fields → Missing, entryline

Software Development

Single Responsibility Principle (SRP) implementation:

Don’t Repeat Yourself (DRY) implementation: