Project Description

We want to develop an application for playing music on a mobile phone or a similar device. The application has three classes – a Playlist class, a Track class and a Driver class. The following diagram shows how the application will be structured.

[image here - see specification file]

The Track class has already been created and the file Track.java is available on the Sulis module site in the Assignment folder.

We want you to create the code for the other two classes (i.e. Driver and Playlist).

Track Class Summary (Code Provided)

The Track class stores the details of a single track as follows

- **A String for the track title**
- **A String for the name of the artist**
- **An int for the year the track was released**
- **An int for the track duration (in seconds)**

The Track provides two constructor(s). One that creates an instance of the Track class using just the title and artist (the other two instance variables are set to 0). A second that creates an instance of the Track class using all four data items (i.e. the title, artist, year and duration).

The class also provides set and get methods for each instance variable in the class.

The class includes a toString method to provide a neat display style for Track instances.

Finally, the class includes an implement Comparable header and a compareTo method that provides a natural ordering for Track instances using the Track Title (case insensitive).

Playlist Class

The Playlist class will be used to store a list of the tracks that the user wants to play. At any time the number of tracks in a playlist can be zero (the playlist is empty) or some number greater than zero. In other words, there is no limit on the number of tracks the playlist can contain. Each playlist will store the following information (i.e. instance values)

- **A String for the name of the playlist**
- **A collection of Track objects. You have to decide what type of collection structure to use to store the tracks.**

The Playlist class should support/provide the following operations/methods

- **Two constructor methods that can be used to create an empty Playlist. The method headers are as follows**

```
// Create a PlayList with a default name (e.g. My PlayList)
public PlayList()
// Create a PlayList with the specified name
public PlayList(String playListName)
```

- **A toString method that can be used to return the contents of PlayList in a readable format. The method header is as follows**

```
public String toString()
```

- **A pair of methods to allow the PlayList name to be inspected (get) or modified (set) with the following headers**

```
public void setName(String name) // Allows name change
public String getName() // Returns current name
```

- **An add method to allow a new track to be added to the list. Tracks are ALWAYS added to the end of the list. The add method header is overloaded as follows**

```
// Add a Track where only the title and artist are known
// The year and duration should be set to zero
public void add(String title, String artist)
// Add a Track where ALL of the data is known
public void add(String title, String artist,
                        int year, int duration)
// Add a previously created instance of the Track class
public void add(Track t)
```

- **A remove method to allow tracks to be deleted from the playlist. The remove header is as follows**

```
public boolean remove(String title)
```

The method is passed the title of the track to be removed.

If there is a track in the playlist with the same title the track should be removed and the method returns true. You should use a case-insensitive comparison when matching track titles.

If there is no track in the playlist with the specified title the method should return false and the list should be left unchanged.

- **A showList method that displays the playlist on the screen in sequential order (i.e. in the order that the tracks appear in the list). The showList header is as follows**

```
public void showList()
```

If there are no tracks in the list the method should display the message "The list is empty".

- **A playAll method that plays all the tracks in the list either in sequence or randomly.**

Passing the value false as a parameter plays the tracks sequentially (i.e. in the sequence they appear in the list).

Passing the value true as a parameter plays the tracks randomly. When playing tracks randomly each track should be played only once. In addition, playing the tracks randomly should NOT alter the sequence of the tracks in the list.

The method header is as follows

```
public void playAll(boolean random)
```

- **A playOnly method that plays tracks in the list that satisfy specified criteria. The playOnly method is overloaded as follows**

```
public void playOnly(String artist)
public void playOnly(int year)
```

The first method will play all the tracks in the list that contain (i.e. anywhere – at the start, in the middle, at the end) the specified text (case insensitive) in the artist name. For example, suppose some of the tracks in a playlist have the artist name "The Eagles." Using `playOnly("eagles")` or `playOnly("EAGLES")` or `playOnly("THE eagles")` should play all the tracks in the list that have the specified strings.

The second method will play all tracks in the playlist that have the specified year as their year of release.

The tracks should be played in the sequence they appear in the list.

If none of the tracks match the specified value then no output should be produced.

We can't actually play the tracks so for testing purposes displaying the track details on the screen will be considered equivalent to playing the track.

You may find it useful to develop additional "helper" methods that might simplify or improve the efficiency of some operations, or might reduce the amount of code you have to write. ALL helper methods should be private.

Driver Class

The Driver class will be used to test the other classes and will contain just a `main` method that uses ALL of the methods provided in the other two classes to show that they have been tested and work correctly. A sample Driver class called `PlayListDriver.java` will be provided shortly.

Submission Requirements

Your solution to the assignment should be submitted on Sulis on or before 16h00 Monday 21st March 2022.

You should submit TWO files (1) PlayList.java that implements the operations specified for the class, and (2) PlayListDriver.java that tests the various operations to verify that they work correctly. In each file you should include prominent comments that contain your ID number and your Name. Sulis will accept any filenames you use but it is extremely important that you adhere to the file name conventions.

You should NOT submit the *Track.java* file. The Track.java provided on the Sulis module site will be used to test your code so please ensure that your PlayList.java code is compatible with the Track.java file provided as part of the assessment resources.

You should NOT submit zip files.

This part of the assignment is worth 25% of the total of 100%.

The marks will be allocated as follows

Component	Allocation
Playlist Constructors	2
toString	1
add methods	3
Remove method	4
showList method	2

playAll method	**9**
Driver program	**4**
Total	**25**