CSC212 2023:	Practical 1 Term 2
Due date:	21 April 2023, 10 PM
Lecturer:	Mr. C. K. Baker
Total:	100 marks

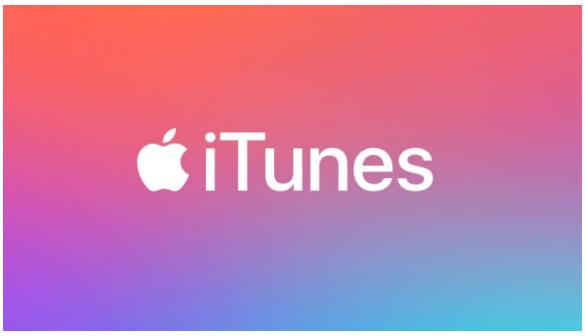
UNIVERSITY of the WESTERN CAPE

Instructions

This practical will test and apply your knowledge of binary search trees. You are required to program in java and use object-oriented programming concepts. Do not use built-in java methods for your tasks: write your own code. Save all your files in a compressed (.zip) folder with the following naming convention **XXYYZZZ_CSC211_2023_Practical_1_Term_2.zip** where **XXYYZZZ** corresponds to your student number. Remember to include the following information at the beginning of each java file:

```
// Surname:        <Your Surname>
// Name:           <Your Name>
// Student no:     <XXYYZZZ>
// Course:         CSC211
// Year:           2023
// Assignment:     Practical 1 Term 2
// File:           <example.java>
```

(leave a line open between this header and your first line of code)



iTunes is one of several platforms that allow users to stream music for a monthly subscription fee. A user typically organises their favourite songs in groupings called playlists. Your task is to efficiently implement such a playlist using a binary search tree.

Question 1 [20 marks]

The following is an interface for a `class Track`:

```
class Track
{
	/* attributes */
	String id; // track ID
	String title; // track title
	String artist; // artist name
	String album; // album title
	int minutes; // minute component of track length
	int seconds; // seconds component of track length
	Track left; // left child
	Track right; // right child

	/* methods */
	String toString();
}
```

1. Create a `class Track` and write code for the following:

   1.1 Encapsulate/hide each attribute by adding an appropriate modifier to its definition

   1.2 A default constructor

   1.3 A loaded constructor

   1.4 Accessor methods for each attribute

   1.5 Mutator methods for each attribute

   1.6 A `toString()` method returns a formatted `String` representation of each attribute for a given track:

   e.g. “Track title: Easy on Me

Artist name: Adele
Album title: 30
Track time: 3 min 45 sec”

Question 2 [40 marks]

The following is an interface for a `class Playlist`:

```
class Playlist
{
      /* attributes */
      Track root; // root of the playlist
      int size; // number of tracks in the playlist

      /* methods */
      void clear();
      Boolean search(String title);
      void insert(String title, String artist, String album, int minutes,
                  int seconds);
      void delete(String title);
      void displayInOrder(Track t);
      int height(Track t);
}
```

2. Create a `class Playlist` and write code for the following:

    2.1 Encapsulate/hide each attribute by adding an appropriate modifier to its definition

    2.2 A default constructor that creates an initially empty tree and initialises each attribute to some default/null value

    2.3 Accessor methods for each attribute

    2.4 Mutator methods for each attribute

    2.5 A method `void clear( )` that makes the tree empty and resets the `size` attribute

2.6 A method `Boolean search(String title)` that returns a `Boolean` value indicating whether a given track is in the playlist or not

2.7 A method `void insert(String title, String artist, String album, int minutes, int seconds)` that creates a new track and adds it to the playlist, using the attribute `title` as the key. The playlist size must be updated.

2.8 A method `void delete(String title)` that removes a track from the playlist. The playlist size must be updated.

2.9 A method `void displayInOrder(Track t)` that prints the inorder traversal of the playlist using `toString()` defined in 1.5

2.10 A method `int height(Track t)` that returns the height of a track in the playlist

Question 3 [40 marks]

Now that you have implemented a binary search tree structure to manage tracks in a playlist, test that your code works correctly.

Execute the driver program provided in `Main.java`. Do not modify the driver code.

The output must produce the correct values for 14 test cases, i.e. 14 size and height values, and the inorder traversal of the final playlist.

Submission and Grading

Include only the following files in your submission folder:

- `Track.java`
- `Playlist.java`

Finally, review the attached rubric for the mark allocation per question.
