L07

Due by the end of Friday, March 10

Today's Activities

In today's lab you will be working with the following concepts:

1. Overloading constructors
2. Overriding toString method
3. Static variables
4. Instance variables

This lab is a continuation of L06 in which you created a class to store some basic information about music tracks, `MusicTrack`.

Download the provided Java file, `MusicTrackDriver` (note that this file is different from the one provided in `L06`) and put it in a new Netbeans project, `L07`. `MusicTrackDriver` is the main class for this lab, and you will only be deleting lines from it. **You won't submit this file!**

Now, create a new java class called `MusicTrack,` which is the file that you will be modifying and submitting at the end. Replace the file you just created with the one you submitted for `L06`. Alternately, you can copy the content of the file you submitted for `L06` to replace the content of the new file. If you put the files into the non-default package, you will have to add the appropriate package statements (You probably need to *change the name of the imported package from l06 to l07).*

Before starting Activity 1, run the project and observe the following output

```
Creating a new record for a music track
-----------------------

MIDNIGHT MOOD
by Michael Brecker
Length: 6.38 minutes
Number of downloads: 2560174
```

Activity #1

Delete the line in **MusicTrackDriver.java** which corresponds to this activity.

Create a constant for **artist**. That would be:

```
public static final String DEF_ARTIST = "unknown";
```

Next, create another constructor for **MusicTrack** which takes a track's title as well as its length and assumes that its artist is unknown (use the constant you just created) and it hasn't been downloaded yet (the `downloadCount` is 0).

Run the program and observe the following output:

```
---[ Activity 1 ]---
The default artist is: unknown
COMA
by unknown
Length: 10.27 minutes
Number of downloads: 0
```

Activity #2

Right after where you declared the instance variables, create a new instance variable `trackID` which is unique for each music track and once set won't change anymore. It's called an *instance constant*. If you cannot remember how to declare it, you will need to go back and check the lecture.

The ID for each track should be set in the primary constructor and must be one greater than the last created track. Use a **static int** in your implementation to declare a *class variable*. The use of the Java keyword **static** is of great importance. You may have 4 different objects of type **MusicTrack**, meaning 4 different areas of memory in RAM are dedicated to those 4 objects. The use of static forces Java to use only one memory location for the variable/constant. *All objects go to that one memory location to get the value.*

To create a class variable for the track ID generator, you need:

```
private static int trackIDGenerator = 1; // first track
will have a trackID of 1
```

A *constant* declaration includes the word **final**. A *class variable* declaration does not. One you cannot change, and the other is a variable. *Just like all objects go to that one memory location to get the value of a constant, all objects go to one memory location to get or change the value of a class variable.* That is because once again the keyword **static** is used in the declaration. Note also that the value of a class variable persists for as long as your program is running. One object can change its value, and another object will see the change, because all objects are using the same variable, because it is static. Here, you can use a static int to ensure that no two tracks have the same ID. Essentially, it will act as a count for the number of tracks which have been created.

Delete the line in **MusicTrackDriver.java** which corresponds to this activity. Run the program and you should get the following output:

```
---[ Activity 2 ]---
3
4
5
6
7
8
9
10
11
12
```

Activity #3

Delete the line in **MusicTrackDriver.java** which corresponds to this activity.

Observe the output as something similar to:

```
---[ Activity 3 ]---
l07.MusicTrack@6ce253f1

l07.MusicTrack@53d8d10a
```

That's not a very useful way to print a track! What is the title? Who is the artist? What is their ID or length? How many times has it been downloaded? Implement the appropriate overriding method to **return a string** that represents the track.

The track has four lines of text

- The first line is the track ID (6 letters, with leading 0's if needed) followed by the title in CAPITAL letters.
- The second line is the artist name (please see the sample output below).
- The third line is the length
- The fourth line is the number of times it has been downloaded.

```
---[ Activity 3 ]---
000001 - MIDNIGHT MOOD
by Michael Brecker
Length: 6.38 minutes
Number of downloads: 2560174

000013 - SOMEBODY TO LOVE
by Queen
Length: 5.16 minutes
Number of downloads: 729515997
```

Note: You can include a leading zero on a number by putting a zero in front of the size of the field. For example `String.format("%04d" , 63)` will display 63 with four digits as `0063`.

Submit this/these files:

- MusicTrack.java

You will be graded on the following:

1. The code is properly formatted and has meaningful variable names and comments where necessary
2. Your @author information as well as class description is found in the javadoc of MusicTrack.java
3. MusicTrack.java compiles and runs without any runtime exceptions
4. A constant has been declared for artist
5. MusicTrack has an overloaded constructor that creates an account with 0 downloadCount and unknown artist
6. ... the constructor has the appropriate javadoc comment
7. ... and this constructor calls the primary constructor (no code duplication)
8. Each MusicTrack gets a unique ID
9. ... which is implemented using a static integer
10. ... and subsequent tracks get neighboring track IDs
11. MusicTrack can be printed easily because of the toString override method
12. ... the first line is track ID and title in the right format
13. ... … the track ID is 6 numbers long and title comes in CAPITAL letters
14. ... the third and fourth lines are representing track's length and downloadedCount