**Reviewing Assignment**

Lab Assignment 7

|  |  |
| --- | --- |
| Started: | Nov 6, 2014 10:34 PM |
| Finished: | Nov 13, 2014 1:50 PM - late |

 1 of 1

**Lab Assignment 7     Total Grade: 19   (of possible 20 points)**

**Score: 19   (of possible 20 points)**

**Assignment 7 - ITunes Objects**

Select one option from below.  All (both) options are worth the same number of points.  The more advanced option(s) are provided for students who find the basic one too easy and want more of a challenge.   Make sure you have read and understood

* both ***modules A*** and ***B*** this week, and
* ***module 2R - Lab Homework Requirements***

before submitting this assignment. Hand in only one program, please.

OPTION A (Basic) ITunes

Understand the Application

An ***ITunes entry*** in music library is a *descriptor* that summarizes information about the tune that it describes.  (It is not the actual tune, which is contained in a large music data file.)  For each ***ITunes entry*** in your own music library file (which is in XML format, but we won't go there)  there are between 20 and 50 fields -- i.e., members.  If you were to look at the file on your system, you would find that the fields have names like ***Genre, Track ID,  Name, Artist, Bit Rate, Sample Rate, Track Type,*** and so on.

We will consider a *simplified* **ITunes** class, that stores these ***ITunes entry*** objects, with only the following four members for each object:

* ***Name*** (the title of the song)
* ***Artist*** (the performing musician or group)
* ***Bit Rate*** (the number of *kilobits* per second that the **ITunes** object streams at or plays at -- higher for better quality, lower for smaller data file size)
* ***Total Time*** (the playing time of the **ITunes** object, represented in *milliseconds*, i.e., 36500 would stand for 36.5 seconds).

The assignment is to first create a class called **ITunes** that represents these four items, and provides solid protection for the fields. Then, the client, **main()**, will instantiate between four or more **ITunes** objects (some using default constructors and some using parameter-taking constructors), display them immediately, mutate many of their fields, display them again to reflect the changes, and finally reset each object to default values and display them one last time. Also, it should do some explicit accessor and mutator tests. This is described in detail further down.

The Program Spec

**Class ITunes Spec**

Private class instance members:

* **String name** - the title of the song.  All legal strings should be between 1 and 128 characters.
* **String artist** - the performing musician or group.  All legal strings should be between 1 and 128 characters.
* **int bitrate** - the number of *kilobits* per second.  A legal value should be between 64 and 705.
* **int totalTime** - the playing time in *milliseconds.*A legal value should be between *MIN\_PLAY\_TIME (5 seconds)* and *1 hour* (expressed in milliseconds)

As stated in the modules, we never want to see a literal in our methods.  So the class should have static members to hold values for the limits described above, as well as default values for any field that is constructed using illegal arguments from the client.  These are put in the **public static** section.

Public class static constants (finals):

* **MIN\_BITRATE** = 64
* **MAX\_BITRATE** = 705
* **MIN\_STR\_LENGTH** = 1
* **MAX\_STR\_LENGTH** = 128
* **MIN\_PLAY\_TIME** = 5000
* **MAX\_PLAY\_TIME** = 1000\*60\*60
* **DEFAULT\_BITRATE** = 64
* **DEFAULT\_PLAY\_TIME** = 5000
* **DEFAULT\_STRING** = " (undefined) "

You should supply all of the following ***public instance methods***:

* ***Constructors*** -- one that takes no parameters (sets values to default) and one that takes all four parameters.
* ***Accessors*** (**get()**s) and ***Mutators*** (**set()**s) for each instance member.
* **String toString()** - a method that returns a **String** which contains all the information of the **ITunes** object.  This **String** can be in any format as long as it is understandable and clearly formatted.  Two of the many possible formats for such a String for one **ITunes** object might be:
  + "Janiva Magness, You Were Never Mine, 276 seconds, 128k bits per second"
  + "Title: You Were Never Mine / Artist: Janiva Magness / Playing Time: 4 minutes 36 seconds / Bit Rate: 128k"
* **void setDefaults()** - a method that can be called either by the client or as a helper method by the constructors.  This method will reset all private data to their default values.
* **void display()** - an output method that sends the **String** returned by the **toString()** to the screen.  **display()** should make use of **toString()** and not send the individual members directly to the screen manually;  instead, it would send the return value of **toString()** to the screen.  However, it can call upon **toString()**but***prepend*** and ***append*** extra formatting for the console.

**ITunes**should be a class distinct from (and not contained within) your main class (which we call **Foothill**). However, you can and should defined it as a non-public class so it can reside in the same file, **Foothill.java**.

**The Foothill main()**

**main()** should instantiate four or more **ITunes** objects, some of them using the default constructor, some using the constructor that takes parameters.  It should immediately display all objects.  Next, it should mutate one or more members of every object, after which it should display all objects a second time.  Then, it should reset all objects to their default values using the appropriate method and display all objects one last time. It does not have to take any input from the user.

**Other Testing**

After the main section, confirm that your mutators correctly filter out and report *bad arguments* to the client by placing a couple mutator calls (not all of them) in *if statements* which print different messages depending on whether the mutator succeeded or failed.  Also, demonstrate at least two different accessor calls to show that they work from **main().**

I am not supplying a sample spec this week -- the above description is adequate..

OPTION B (Intermediate) Arrays and More

Add a function **timeInMinutesAndSeconds()** that returns a **String** in the form  show above, .e.g., "4 minutes 36 seconds".

In your client, instantiate ***an array*** of 10 **ITunes** objects with values obtained either from the user or hard-coded by a clever use of helper **String** and **int** arrays.   Use loops, not a long list of statements that reference literal indexes.  We never want to reference literal indexes in any program that uses arrays.

**Answer**

* text/plain[foothillAssignment7.txt](https://myetudes.org/access/mneme/content/private/mneme/cff3240c-b51c-41f6-80dc-4db4530bdd05/submissions/15103539/cafb7225-d301-46d6-80f2-390b1edd0a61/foothillAssignment7.txt)

[[https://myetudes.org/ambrosia_library/icons/collapse.gif](https://myetudes.org/portal/tool/09d2d876-2329-4a14-000d-b3da1e731165/review/15103539/list) Model Answer](https://myetudes.org/portal/tool/09d2d876-2329-4a14-000d-b3da1e731165/review/15103539/list)

/\* CS 1A Lab 7

 \* Instructor Solution

 \*/

public class Foothill

{

   public static void main( String[] args )

   {

      iTunes tune1 = new iTunes(), tune2 = new iTunes(),

            tune3 = new iTunes("Hobo Blues", "John Lee Hooker", 128, 182000),

            tune4 = new iTunes("Give It All U Got", "Lil Jon", 128, 218000);

      tune1.display();

      tune2.display();

      tune3.display();

      tune4.display();

      // mutate tune1:

      tune1.setArtist("Steely Dan");

      tune1.setName("Black Cow");

      tune1.setBitRate(256);

      tune1.setTotalTime(310 \* 1000);  // 310 seconds

      // mutate others:

      tune2.setBitRate(512);

      tune3.setBitRate(512);

      tune4.setBitRate(512);

      System.out.print("\nAll tunes after mutation");

      tune1.display();

      tune2.display();

      tune3.display();

      tune4.display();

       // reset to defaults and test

      tune1.setDefaults();

      tune2.setDefaults();

      tune3.setDefaults();

      tune4.setDefaults();

      System.out.println("\nsetDefault() Tests ---------- ");

      tune1.display();

      tune2.display();

      tune3.display();

      tune4.display();

      System.out.println("\nMutator Tests ---------- ");

      if (!tune2.setArtist(""))

         System.out.println("\n Correctly rejected blank string");

      if (!tune2.setBitRate(999))

         System.out.println("\n Correctly rejected out-of-range bit rate");

      System.out.println("\nAccessor Tests ---------- ");

      System.out.println("tune1 artist: " + tune1.getArtist() );

      System.out.println("tune3 total time (ms): " + tune3.getTotalTime() );

   }

}

// our target class, iTunes

class iTunes

{

   private String name;

   private String artist;

   private int bitrate;

   private int totalTime;

   static final int MIN\_BITRATE = 64;

   static final int MAX\_BITRATE = 705;

   static final int MIN\_STR\_LENGTH = 1;

   static final int MAX\_STR\_LENGTH = 128;

   static final int MIN\_PLAY\_TIME = 1000 \* 5;    // five seconds

   static final int MAX\_PLAY\_TIME = 1000\*60\*60;  // one hour

   static final int DEFAULT\_BITRATE = 64;

   static final int DEFAULT\_PLAY\_TIME = MIN\_PLAY\_TIME;

   static final String DEFAULT\_STRING = " (undefined) ";

   public iTunes()

   {

      setDefaults();

   }

   public iTunes(String nm, String art, int btrt, int tTime)

   {

      if ( !setName(nm) )

         name = DEFAULT\_STRING;

      if ( !setArtist(art) )

         artist = DEFAULT\_STRING;

      if ( !setBitRate(btrt) )

         bitrate = DEFAULT\_BITRATE;

      if ( !setTotalTime(tTime) )

         totalTime = DEFAULT\_PLAY\_TIME;

   }

   // accessors - ok to use brief notation for accessors

   public String getName() { return name; }

   public String getArtist() { return artist; }

   public int getBitrate() { return bitrate; }

   public int getTotalTime()  { return totalTime; }

   public boolean setName(String nm)

   {

      if (nm.length() < MIN\_STR\_LENGTH || nm.length() > MAX\_STR\_LENGTH)

         return false;

      name = nm;

      return true;

   }

   public boolean setArtist(String art)

   {

      if (art.length() < MIN\_STR\_LENGTH || art.length() > MAX\_STR\_LENGTH)

         return false;

      artist = art;

      return true;

   }

   public boolean setBitRate(int btrt)

   {

      if (btrt < MIN\_BITRATE || btrt > MAX\_BITRATE)

         return false;

      bitrate = btrt;

      return true;

   }

   public boolean setTotalTime(int tTime)

   {

      if (tTime < MIN\_PLAY\_TIME || tTime > MAX\_PLAY\_TIME)

         return false;

      totalTime = tTime;

      return true;

   }

   public String toString()

   {

      String retVal;

      retVal = name + "\", by " + artist

            + "\n Duration: " + totalTime/1000

            + " seconds, Bit Rate: " + bitrate;

      return retVal;

   }

   public void display()

   {

      System.out.println("\niTunes Song ---------:\n \""

            + toString());

   }

   public void setDefaults()

   {

      name = DEFAULT\_STRING;

      artist = DEFAULT\_STRING;

      totalTime = DEFAULT\_PLAY\_TIME;

      bitrate = DEFAULT\_BITRATE;

   }

}

/\* ------------------------- sample run -----------------------------

iTunes Song ---------:

 " (undefined) ", by  (undefined)

 Duration: 5 seconds, Bit Rate: 64

iTunes Song ---------:

 " (undefined) ", by  (undefined)

 Duration: 5 seconds, Bit Rate: 64

iTunes Song ---------:

 "Hobo Blues", by John Lee Hooker

 Duration: 182 seconds, Bit Rate: 128

iTunes Song ---------:

 "Give It All U Got", by Lil Jon

 Duration: 218 seconds, Bit Rate: 128

All tunes after mutation

iTunes Song ---------:

 "Black Cow", by Steely Dan

 Duration: 310 seconds, Bit Rate: 256

iTunes Song ---------:

 " (undefined) ", by  (undefined)

 Duration: 5 seconds, Bit Rate: 512

iTunes Song ---------:

 "Hobo Blues", by John Lee Hooker

 Duration: 182 seconds, Bit Rate: 512

iTunes Song ---------:

 "Give It All U Got", by Lil Jon

 Duration: 218 seconds, Bit Rate: 512

setDefault() Tests ----------

iTunes Song ---------:

 " (undefined) ", by  (undefined)

 Duration: 5 seconds, Bit Rate: 64

iTunes Song ---------:

 " (undefined) ", by  (undefined)

 Duration: 5 seconds, Bit Rate: 64

iTunes Song ---------:

 " (undefined) ", by  (undefined)

 Duration: 5 seconds, Bit Rate: 64

iTunes Song ---------:

 " (undefined) ", by  (undefined)

 Duration: 5 seconds, Bit Rate: 64

Mutator Tests ----------

 Correctly rejected blank string

 Correctly rejected out-of-range bit rate

Accessor Tests ----------

tune1 artist:  (undefined)

tune3 total time (ms): 5000

------------------------------------------------------------------- \*/

**Comments**

Nice work, Dmitri.

CLASS  
\* Your data is correctly labeled private  
\* Static constants are correctly declared and initialized.  
\* Your methods all have the signatures given in the spec. Following a spec to the letter is crucial for being able to program in groups.  
  
MAIN/RUN  
\* Your client instantiates enough iTunes objects, mutates and displays them.  
\* The mutators and accessors are tested well and your output is descriptive.  
\* You have tested both constructors: good.  
  
CONSTRUCTORS  
\* Both constructors are well designed.  
  
ACCESSORS & MUTATORS  
\* You have provided all of the required get() and set() methods for your class data.  
\* The mutators are filtering data well so that invalid data cannot be set.  
\* You are using the constant values to replace literal strings and numbers. This makes your code easier to understand and less prone to errors.  
  
TOSTRING() & DISPLAY  
\* Your toString() method returns a nicely formatted string with all needed information.  
\* display() makes good use of toString() and does a nice job of displaying the iTunes data to the console.  
  
FORMATTING/STYLE  
\* Your indentation is good. The code and output are easy to read.

1 day late (-1)

 1 of 1

public class Foothill

{

public static void main(String[] args)

{

iTunes songOne;

iTunes songTwo;

iTunes songThree;

iTunes songFour;

iTunes songFive;

songOne = new iTunes("On the Floor", "JLo", 320, 360000);

songTwo = new iTunes("Poly", "Nirvana", 200, 49500);

songThree = new iTunes();

songFour = new iTunes("Moon Sonate", "Beethoven", 128, 500000);

songFive = new iTunes();

System.out.println("Song One");

songOne.display();

if (!songOne.setName("Floor Song"))

printErrorMessage("Title");

if (!songOne.setBitrate(224))

printErrorMessage("Bitrate");

System.out.println("Changed Song:");

songOne.display();

songOne.setDefaults();

System.out.println("Song after reset:");

songOne.display();

System.out.println("");

System.out.println("Song Two");

songTwo.display();

if (!songTwo.setName("School"))

printErrorMessage("Title");

if (!songTwo.setArtist("Curt Cobain"))

printErrorMessage("Artist");

System.out.println("Changed Song:");

songTwo.display();

songTwo.setDefaults();

System.out.println("Song after reset:");

songTwo.display();

System.out.println("");

System.out.println("Song Three");

songThree.display();

if (!songThree.setBitrate(128))

printErrorMessage("Bitrate");

if (!songThree.setName("Moon"))

printErrorMessage("Title");

System.out.println("Changed Song:");

songThree.display();

songThree.setDefaults();

System.out.println("Song after reset:");

songThree.display();

System.out.println("");

System.out.println("Song Four");

songFour.display();

if (!songFour.setTotalTime(325))

printErrorMessage("Total time");

if (!songFour.setName("Sonata"))

printErrorMessage("Title");

System.out.println("Changed Song:");

songFour.display();

songFour.setDefaults();

System.out.println("Song after reset:");

songFour.display();

System.out.println("");

System.out.println("Song Five");

songFive.display();

if (!songFive.setArtist("oiasdlkadlkb,bdb asdnf,adsak';k'ejmnsd,mnmdsnm.,"

+ "sassdl;as';ihsf;dnbdfsjoiero[hbbreoherohfsdb,mdsf,.dfnireioerret"

+ "kdfjhsufgreufgiwruyiulfihdkghktuhgdotirgdhrtilghorihgeoihgjli"))

printErrorMessage("Artist");

System.out.println("Changed Song:");

songFive.display();

songFive.setDefaults();

System.out.println("Song after reset:");

songFive.display();

System.out.println("");

if (!songThree.setName("March"))

printErrorMessage("Title");

System.out.println("Song title: " + songThree.getName());

if (!songFour.setBitrate(300000000))

printErrorMessage("Bitrate");

System.out.println("Song bitrate: " + songFour.getBitrate()+"k");

if (!songFive.setTotalTime(328000))

printErrorMessage("Total time");

System.out.println("Song total time: " + songFive.getTotatTime());

}

private static void printErrorMessage(String paramName)

{

System.out.println(paramName + " had not been changed");

}

}

class iTunes

{

private String name;

private String artist;

private int bitRate;

private int totalTime;

public static final int MIN\_BITRATE = 64;

public static final int MAX\_BITRATE = 705;

public static final int MIN\_STR\_LENGTH = 1;

public static final int MAX\_STR\_LENGTH = 128;

public static final int MIN\_PLAY\_TIME = 5000;

public static final int MAX\_PLAY\_TIME = 1000\*60\*60;

public static final int DEFAULT\_BITRATE = 64;

public static final int DEFAULT\_PLAY\_TIME = 5000;

public static final String DEFAULT\_STRING = "(undefined)";

iTunes()

{

setDefaults();

}

iTunes(String name, String artist, int bitRate, int totalTime)

{

if (!setName(name))

this.name = DEFAULT\_STRING;

if (!setArtist(artist))

this.artist = DEFAULT\_STRING;

if (!setBitrate(bitRate))

this.bitRate = DEFAULT\_BITRATE;

if (!setTotalTime(totalTime))

this.totalTime = DEFAULT\_PLAY\_TIME;

}

public boolean setName(String name)

{

if(name.length() > MAX\_STR\_LENGTH || name.length() < MIN\_STR\_LENGTH)

return false;

this.name = name;

return true;

}

public boolean setArtist(String artist)

{

if(artist.length() > MAX\_STR\_LENGTH || artist.length() < MIN\_STR\_LENGTH)

return false;

this.artist = artist;

return true;

}

public boolean setBitrate(int bitRate)

{

if (bitRate > MAX\_BITRATE || bitRate < MIN\_BITRATE)

return false;

this.bitRate = bitRate;

return true;

}

public boolean setTotalTime( int totalTime)

{

if (totalTime > MAX\_PLAY\_TIME || totalTime < MIN\_PLAY\_TIME)

return false;

this.totalTime = totalTime;

return true;

}

public void setDefaults()

{

this.name = DEFAULT\_STRING;

this.artist = DEFAULT\_STRING;

this.bitRate = DEFAULT\_BITRATE;

this.totalTime = DEFAULT\_PLAY\_TIME;

}

public String toString()

{

return "Title: " + name + " " + "Artist: " + artist + " " + "Bitrate: "

+ bitRate + "k " + "Time(ms): " + totalTime ;

}

public void display()

{

System.out.println(toString());

}

public String getName()

{

return this.name;

}

public String getArtist()

{

return this.artist;

}

public int getBitrate()

{

return bitRate;

}

public int getTotatTime()

{

return totalTime;

}

}

/\*----------paste of run from console window------------

Song One

Title: On the Floor Artist: JLo Bitrate: 320k Time(ms): 360000

Changed Song:

Title: Floor Song Artist: JLo Bitrate: 224k Time(ms): 360000

Song after reset:

Title: (undefined) Artist: (undefined) Bitrate: 64k Time(ms): 5000

Song Two

Title: Poly Artist: Nirvana Bitrate: 200k Time(ms): 49500

Changed Song:

Title: School Artist: Curt Cobain Bitrate: 200k Time(ms): 49500

Song after reset:

Title: (undefined) Artist: (undefined) Bitrate: 64k Time(ms): 5000

Song Three

Title: (undefined) Artist: (undefined) Bitrate: 64k Time(ms): 5000

Changed Song:

Title: Moon Artist: (undefined) Bitrate: 128k Time(ms): 5000

Song after reset:

Title: (undefined) Artist: (undefined) Bitrate: 64k Time(ms): 5000

Song Four

Title: Moon Sonate Artist: Beethoven Bitrate: 128k Time(ms): 500000

Total time had not been changed

Changed Song:

Title: Sonata Artist: Beethoven Bitrate: 128k Time(ms): 500000

Song after reset:

Title: (undefined) Artist: (undefined) Bitrate: 64k Time(ms): 5000

Song Five

Title: (undefined) Artist: (undefined) Bitrate: 64k Time(ms): 5000

Artist had not been changed

Changed Song:

Title: (undefined) Artist: (undefined) Bitrate: 64k Time(ms): 5000

Song after reset:

Title: (undefined) Artist: (undefined) Bitrate: 64k Time(ms): 5000

Song title: March

Bitrate had not been changed

Song bitrate: 64k

Song total time: 328000

----------------------------------------------------- \*/