**Project 4: MP3 Library Manager**

Due: July 28, 2010

In this project you will improve on your MP3 File Library developed in Project 3.  The improvements include the addition of sorting and searching of MP3 files and a better user interface.  As before, your program should be able to play an mp3 file and display some of the embedded tags in an MP3 file, such as title, author, album, and data.  It should be possible to sort and search your MP3 files by any of the tags.

**As before**, **this project involves dealing with mp3 files, and you must only use *legally* obtained mp3 files.**

In this project, you are also required to extend your existing JavaDoc comments for all of your classes.

Submit your project directory to cs1302a on odin, a directory called Project4 (using the submit command):

* All the source code, i.e. .java files. Please do NOT include .class files.  Javadoc comments are required. Of course, regular “program logic” comments are required in your source code.  As usual, your source code should be well formatted.
* A README file telling us how to compile your program and how to execute it.

**Project description:**

1. Your program must be in package edu.uga.cs1302.mp3manager.
2. Add one more public method to your LinkedList class:

Object[] toArray()

The meaning of the above method should be the same as the equivalent method provided by java.util.LinkedList<E>.  Please, consult Java SE API documentation.

1. Modify your implementation of the MP3File class to assign any undefined MP3 tags of *author, title*, and *album* to “Unknown”. Assign an unspecified *date* of an mp3 file to the year 1900.  Also, add the following methods (and a suitable class field):

int     getYear()

void    setYear(int year)

1. Implement the MP3Collection class, which should be based on your MyMP3Files class developed in the previous project.  The new class should represent your collection of MP3Files using your implementation of LinkedList and provide the following public methods:

MP3Collection()

Create an empty collection.

MP3Collection(String directoryPathname)

Create a collection containing files in the specified directory. The

collection should include only files with the .mp3 extension.

1. * + - 1. void    sort(Comparator comp)

Sort the collection according to the given comparator.

1. int     binarySearch(MP3File file, Comparator comp)

Search for a given mp3 file, using the given comparator.  Return the

index of the file in the collection or -1 if not found. The collection

must be sorted using the same comparator.

1. int     size()

Return the size of the collection.

1. MP3File getFile(int index) throws IndexOutOfBoundsException

Return an MP3File at the given index position in the collection.

Throw an IndexOutOfBoundsException exception if the index value

is out of bounds.

1. void    play(int index) throws IndexOutOfBoundsException

Start playing a file at a given index position. Throw an

IndexOutOfBoundsException exception if the index value is out of

bounds.

1. The MP3Collection class should maintain an internal (private) array of references to the MP3File objects on your list (you should use your new toArray() method to establish the content of this array). The methods sort and binarySearch should use the established internal array to perform their functions.  The sort method should be implemented using the *QuickSort* algorithm.  You should create suitable *comparators* for the *author, album, title*, and *date* of an MP3File object.  The *date* comparator should use the *year* field for comparison purposes, instead of the String representation of the date.

5. Create a class MP3CollectionManager which implements a simple a text-oriented manager for an MP3Collection.  The manager implements the following commands:

list

* + - * 1. List the current mp3 file (by showing its mp3 tags).  The current file should be listed along with its predecessor and successor.  If the current file has no predecessor and/or successor files, the text shown should be “top of the list”, or “end of the list”, respectively.

1. next [n]
   1. Advance forward by n files (the default is 1). If the current file has no successor, the text shown should be “end of the list”.
2. prev [n]
   1. Advance backward by n files (the default is 1). If the current file has no predecessor, the text shown should be “top of the list”.
3. play
   1. Start playing the current mp3 file.
4. info
   1. Show the current mp3 file’s mp3 tags.
5. sort [album|author|title|date]

Sort the collection by the given tag.  The sorting should be in ascending order.   *Important*: after the sort, the current file should be maintained. This can be

achieved by saving the reference of the current object and then locating the

same reference in the sorted array. For example,

sort author

* + - 1. sorts the collection according to the *author* tags.

1. find [album|author|title] substring    **This is a BONUS.**

Find the next mp3 file including the given substring in the *album*, *author*, or *title*.  The located file should become the *current* one.  Before searching, the collection is sorted according to the same tag as used in the find comand. If the file is not found, the manager should display the text “Not Found” and the current file should not be changed. The substring starts after the tag and extends till the end of the line. For example,

find title Thriller

locates an mp3 file with the word Thriller in its title, while:

find author Ten Years After

* + 1. locates an mp3 file with the string Ten Years After in its *author* tag.

1. The ordering of the files in the collection should be according to the most recently performed sort command.  That is, the ordering should be according to the ordering of MP3File objects in the internal array, created in the class MP3Collection. Initially, the ordering should be according to the *title*. The manager should maintain the *current* file. Initially, the current file should be the first file in the initial ordering. All reading should be from System.in and writing (excpet for playing the mp3files) to System.out.
2. 6. Run javadoc to create the API documentation for your project in directory html, a subdirectory in your project directory Project3.

**Things to note:**

* **Ethics note:** remember that you should not use *illegally* obtained mp3 files for your listening pleasure, or to test your program.  There are web sites that offer mp3 files for free (for example, [www.acidplanet.com](http://www.acidplanet.com/), or free music downloads from amazon.com).  Many bands and individual musicians offer free samples of their music, as well.  You should use only *legally* obtained music files.Please, consult the legal statement of the site from which you are downloding files. Remember that even thought the files may be free, that may not give you the right to share the files with others!
* All of your classes must be in package edu.uga.cs1302.mp3manager.
* Your must use your modified LinkedList to store MP3File objects.
* You must provide suitable JavaDoc comments for your classes, methods, and constants.

Your design should be reasonably efficient, and in accordance with object oriented design principles (encapsulation, information hiding, inheritance, etc.).